

# Using DAC with HIWARE Tools



**Software Development, Quality and Documentation Tool**



Development Assistant for C V4.0 Documentation  
Release date: September 25, 2002

File name: "DAC HIWARE TN LT.pdf"  
Version 1.8

URL: <http://www.RistanCASE.com/dac/v40/specificsupport/metrowerks.php>

Copyright © 1990-2002 RistanCASE GmbH Switzerland. All rights reserved.

The information contained in this document is subject to change without notice.

RistanCASE GmbH makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. RistanCASE GmbH shall not be liable for errors contained herein, direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

ALL RIGHTS RESERVED. No part of this publication may be copied in any form, by photocopy, microfilm, retrieval system, or by any other means now known or hereafter invented without the prior written permission of RistanCASE GmbH.

IBM PC, XT, AT are trademarks of the International Business Machine Corporation.

Microsoft® Windows is a registered trademark of the Microsoft Corporation.

All other trademarks used in this document may be trademarks or registered trademarks of their respective owners and are hereby acknowledged.

### **Corporate Headquarters**

RistanCASE GmbH  
Zielackerstrasse 19  
CH-8304 Wallisellen-Zurich  
Switzerland

Telephone: +41 (0) 1 883 35 70  
Fax: +41 (0) 1 883 35 74

E-mail: [info@RistanCASE.com](mailto:info@RistanCASE.com)  
Web: [www.RistanCASE.com](http://www.RistanCASE.com)

---

This document was designed to be distributed electronically and then printed on a laser printer on an as-needed basis. Therefore, the fonts and layout of this document have been chosen for optimal printing rather than for optimal viewing on-screen. To view this document on-screen, however, simply increase the magnification using the magnification box at the top of the window. For the best results when viewing dialog boxes on-screen, increase the magnification to 200%. Some figures have hot links on them.

# Contents

<b>I Requirements .....</b>	<b>1</b>
<b>2 Configuring DAC .....</b>	<b>2</b>
2.1 Creating a New Project.....	3
2.2 Configuring Working Directories .....	3
<i>Project Root directory .....</i>	<i>4</i>
<i>Referential Project Root directory .....</i>	<i>4</i>
<i>Header Directories .....</i>	<i>4</i>
<i>Database directory .....</i>	<i>4</i>
<i>User Help file .....</i>	<i>5</i>
2.3 Configuring File Types .....	5
<i>C Source file .....</i>	<i>5</i>
<i>Assembler Source file .....</i>	<i>5</i>
<i>Header file .....</i>	<i>5</i>
<i>Document file.....</i>	<i>6</i>
<i>Text file .....</i>	<i>6</i>
<i>Referential pairs .....</i>	<i>6</i>
2.4 Configuring Analysis for Symbols.....	6
2.4.1 General .....	7
<i>Environment substitutes.....</i>	<i>7</i>
2.4.2 C Source.....	8
<i>Maximum identifier length .....</i>	<i>8</i>
<i>Special table processing .....</i>	<i>8</i>
<i>Defines / Control file .....</i>	<i>8</i>
<i>Nested comments .....</i>	<i>9</i>
<i>C++ comments .....</i>	<i>9</i>
<i>Traditional.....</i>	<i>9</i>
<i>Ignore #line .....</i>	<i>9</i>
2.5 Configuring Compiler Dialect and Header Directories .....	9
<i>Source .....</i>	<i>10</i>
<i>Compiler header directories.....</i>	<i>10</i>
<i>Preinclude header file .....</i>	<i>11</i>

2.6 Configuring Assembler Dialect and Header Directories .....	11
<i>Source</i> .....	11
<i>Assembler header files</i> .....	12
<i>Preinclude header file</i> .....	12
<i>Support for structured types</i> .....	12
<i>MCUasm compatibility</i> .....	12
<i>Case insensitive symbols</i> .....	12
2.7 Adding Files to the Project .....	12
2.8 Building the Database .....	13
<b>3 Integrating the Tools .....</b>	<b>16</b>
3.1 User-Defined Actions (UDA) Setup .....	16
3.2 Makefile Template Setup .....	17
3.3 Using HIWARE Tools by Default .....	19
<b>4 Limitations of Static Code Analysis .....</b>	<b>20</b>
4.1 Analysis for Symbols, C Source .....	20
4.2 Analysis for Symbols, ASM Source .....	20
4.2.1 Notes about assembler support for the specific HIWARE assemblers .....	22
<b>5 Debugger .....</b>	<b>24</b>
5.1 Installation of communication DLL's .....	24
5.2 Configuring DAC Debugger Options .....	25
5.2.1 Debugger .....	26
5.2.2 Connection .....	26
<i>Program to start</i> .....	26
<i>Timeouts</i> .....	27
5.2.3 Project Dependent .....	27
<i>Switches</i> .....	27
<i>Load binary</i> .....	27
<i>Ask before sending to debugger</i> .....	27
<i>Check debugging through non-project files</i> .....	28
5.3 Configuring HI-WAVE Debugger Options .....	28
5.4 Synchronized Debugging .....	29
5.5 Troubleshooting .....	30
<b>6 Index .....</b>	<b>33</b>

---

# I Requirements

---

**NOTE:**

This technical note only provides information on how to configure DAC to facilitate your work with the HIWARE HC05 V5.x Tools, for which the document has primarily been written. Using it as a basis, DAC can easily be configured to work with other HIWARE tools. For further information on DAC, please refer to "Development Assistant for C" documentation V4.0.

- **DAC - V4.0.055 or later** - (Development Assistant for C - RistanCASE). The latest version, with Demo Mode license included, can be downloaded from the following URL:

[http://www.RistanCASE.com/dac/v40/dac\\_download.php](http://www.RistanCASE.com/dac/v40/dac_download.php)

If you are running DAC in Demo Mode, you can easily obtain a trial license and enjoy all the comforts of DAC for two weeks! For more details, choose **Technical Support** from the **Help** menu.

- **GNU make V3.75 or later** - (Free Software Foundation). The gmake.exe with RistanCASE improvements can be downloaded from the following URL:

[http://www.RistanCASE.com/gnu/gnu\\_products.php](http://www.RistanCASE.com/gnu/gnu_products.php)

In the following sections, it is assumed that your HIWARE HC05 V5.x tools have been installed in the "*C:\Hiware\demo\HC05*" folder. In the text that follows, the "*C:\Hiware*" folder will be referred to as the "HIWARE folder." You may have to adapt the paths used in the example provided to match your current installation paths.

HIWARE HC05 V5.x tools and the GNU make utility folders have to be in the system PATH. The GNU make should be named "*gmake.exe*".

It is also assumed that DAC has been installed in the "*C:\Program Files\RistanCASE\Development Assistant for C*" folder, which does not have to be in the system PATH. In the text that follows this folder will be referred to as the "DAC folder."

---

## 2 Configuring DAC

---

To make the most of your HIWARE HC05 V5.x tools with DAC, these are the steps to follow:

- [Creating a New Project](#)
- [Configuring Working Directories](#)
- [Configuring File Types](#)
- [Configuring Analysis for Symbols](#)
- [Configuring Compiler Dialect and Header Directories](#)
- [Configuring Assembler Dialect and Header Directories](#)
- [Adding Files to the Project](#)
- [Building the Database](#)
- [Integrating the Tools](#)

As you progress through this technical note, you will be led through the process of creating a new DAC project named "Demo." The "Demo" example project can be found on RistanCASE's CD-ROM in the "DAC\Technical Notes\HIWARE" folder, or downloaded from RistanCASE's web site: <http://www.RistanCASE.com/dac/v40/specific-support/metrowerks.php>.

Tools supported by DAC - V4.0.055 or later

Supported compilers	Supported assemblers
HIWARE 8051 V5.x	
HIWARE H8/500 V5.x	HIWARE H8/500 V5.x
HIWARE HC05 V5.x	HIWARE HC05 V5.x
HIWARE HC08 V5.x	HIWARE HC08 V5.x

Tools supported by DAC - V4.0.055 or later (Continued)

Supported compilers	Supported assemblers
HIWARE HC11 V5.x	HIWARE HC11 V5.x
HIWARE HC12 V5.x	HIWARE HC12 V5.x
HIWARE HC16 V5.x	HIWARE HC16 V5.x
HIWARE M68xxx/CPU32 V5.x	HIWARE M68xxx/CPU32 V5.x
HIWARE ST7 V5.x	HIWARE ST7 V5.x
HIWARE M-CORE V5.x	HIWARE M-CORE V5.x
HIWARE XA V5.x	HIWARE XA V5.x
Supported debugger	
HI-WAVE V6.0 and later	

## 2.1 Creating a New Project

Start DAC and choose **New Project** from the **Project** menu. Browse through the "`\demo\HC05\Sources`" subfolder of the HIWARE folder and enter the project file name "Demo". A project file will be created. The standard project file extension is ".dcp", so the created file will be named "`Demo.dcp`".

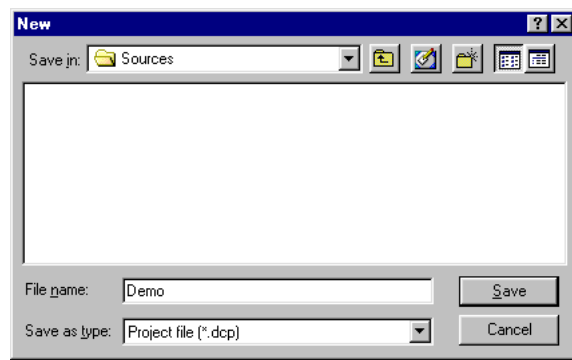


Figure 2.1 Creating a new project

## 2.2 Configuring Working Directories

On the **Options** menu, click **Project** to open the **Project Options** dialog box. Use this dialog box to set project directories:

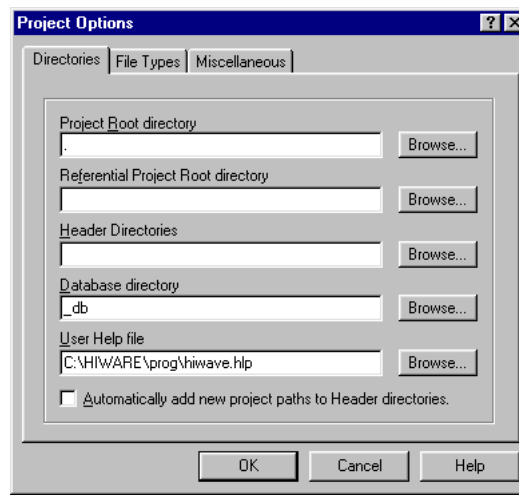


Figure 2.2 Configuring working directories

### Project Root directory

The full path is expected in the Project Root directory text box or, as a special value, a single period character, which stands for "the directory where the project file resides." The names of all files belonging to the project are considered relative to the Project Root directory if the full file path is not given. In our example, keep the single period character for the Project Root directory.

### Referential Project Root directory

If not empty, it specifies an alternative path for searching for files that DAC fails to find in the original project path. DAC will attempt to find files with the referential extension in the Project Root directory prior to searching the referential Project Root directory. The specified path may either be full or relative to the Project Root, and it may not specify a sub-folder in the Project Root directory tree. In this example it should be left empty.

### Header Directories

You should specify paths to all header directories used in the project. The paths are separated by a semicolon.

### Database directory

It allows the Symbols and Software Metrics database files folder to be set. The path in question can be absolute or relative to the Project Root directory. If the Database directory is left unspecified, the Symbols and Software Metrics database files are created in the Project Root directory along with source and other files. It is recommended that you should specify the Database directory, to keep these files from mixing. Enter "\_db" in the box.



## User Help file

It enables you to set the User Help file, for example, the Compiler Help file. The shortcut key for the User Help file can be set in the Keyboard definition file (default CTRL + SHIFT + F1). Browse through the "prog" subfolder of the HIWARE folder and select the "hiwave.hlp" help file. This will give you an entry point for HIWARE Windows Help.

## 2.3 Configuring File Types

### INFO:

When TEXTPATH environment variable is defined, the assembler will store the assembly listings it has produced in the first directory specified there. If TEXTPATH is not set, the generated assembly listings will be stored in the directory where the source file has been found.

On the **Options** menu, click **Project** and select the **File Types** tab. Here you can set file extensions for the basic project file types. A file type extension list consists of up to 10 file extensions separated by a white space or the "." character. To ensure maximally efficient use of HIWARE HC05 V5.x tools, fill in the boxes as shown in the following picture:

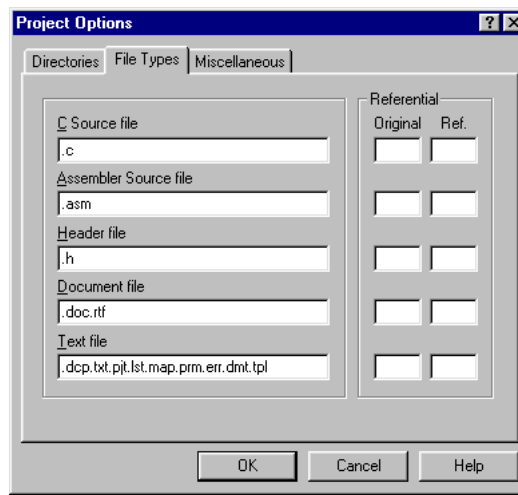


Figure 2.3 Configuring File Types

### C Source file

Files with one of the extensions listed in this box will be considered C source files. Header file extensions should not be defined within this field. Enter ".c" in the box.

### Assembler Source file

Files with one of the extensions listed in this box will be considered included Assembler files. Module file extensions should not be defined within this field. Enter ".asm" in the box.

### Header file

Files with one of the extensions listed in this box will be considered included C header files. Module file extensions should not be defined within this field. Enter ".h" in the box.

**Document file**

Files with one of the extensions listed in this box will be considered document files. Enter ".doc.rtf" in the box.

**Text file**

Files with one of the extensions listed in this box will be considered editable (text) files. If files without extensions are to be considered editable, the list ends with a period ".". Enter ".dcp.txt.pjt.lst.map.prm.err.dmt.tpl" in the box.

**Referential pairs**

Fields for entering alternative (referential) extensions are located in this area. The original extension is entered in the Original column, while the alternative extension is entered in the Referential column. Up to five original / alternative pairs can be set. If opening the file with the original extension fails, DAC will attempt to find and open the file with the identical name and the alternative extension.

## **2.4 Configuring Analysis for Symbols**

For this example you may need to configure Analysis for Symbols Options from [General](#) and [C Source](#) sections.

## 2.4.1 General

On the **Options** menu, click **Analysis for Symbols**, and then click the **General** tab to open the following dialog box:

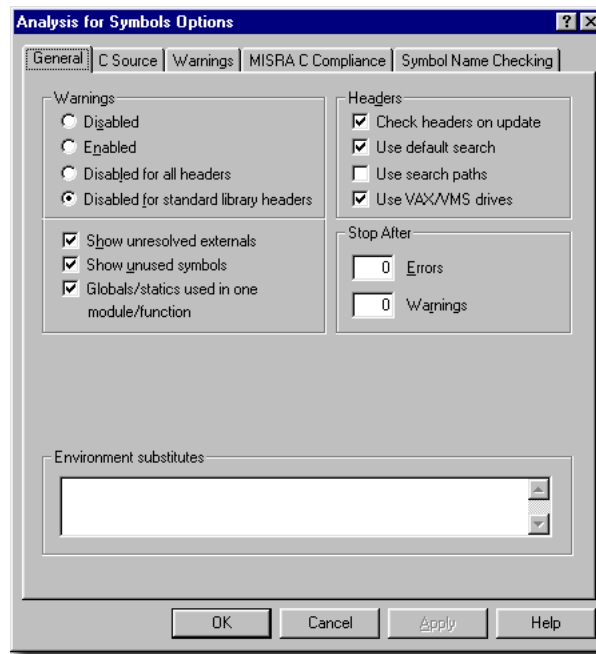


Figure 2.4 General Analysis for Symbols Options

### Environment substitutes

Environment variables defined in Windows can be made a part of various path specifications in DAC by enclosing the variable identifier in "%" characters. Apart from variables defined in the Windows environment, it is possible, in **Environment substitutes**, also to define "DAC environment variables" in the form of:

```
env_var=env_string
```

If the %env\_var% construct is found in DAC paths (project root directory, database directory, include path, %Env(env\_var), ...) %env\_var% will be replaced with env\_string.

These substitutions are also necessary if the environment variables of your development environment have been set only in the DOS window in which you build project binaries. As the variables are not known at Windows level, if you intend to use them, you will have to define them in DAC, in **Environment substitutes**.

For a detailed explanation of other options in the **General** tab please consult DAC help.

### 2.4.2 C Source

On the **Options** menu, click **Analysis for Symbols**, and then click the **C Source** tab to open the following dialog box:

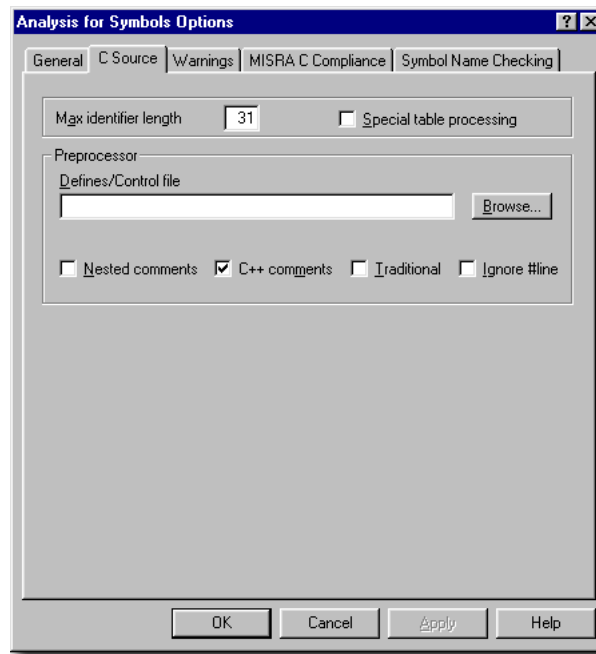


Figure 2.5 C Source Analysis for Symbols Options

#### Maximum identifier length

This is the number of significant characters in the identifiers. The remaining characters are ignored. The default value is 31.

#### Special table processing

It determines the treatment of arrays of pointers to functions in the analysis. If this option is selected, such variables are treated as functions, otherwise, they are treated in the usual way (default).

#### Defines / Control file

It determines the list of macro definitions to be set at the beginning of the module preprocessing. This box may contain the control file name (with absolute or relative path in relation to the project root) with the prefix @. The prefix @ is also used in the Project file ".dcp" to specify included files. Therefore, you should avoid using the character @ at the beginning of the names of files and folders that are to be used in the project. Control file lines beginning with -i or -I subsequently contain the list of directories for searching for #include files, separated by a semicolon. Control file lines beginning with -d or -D subsequently contain the list of macro definitions, as described in the previous paragraph. The remaining control file lines are ignored.

**Nested comments**

It allows proper recognition of nested C style comments, for example `/*  
/* ... */*/`.

**C++ comments**

It determines if C++ comments (beginning with `"/"/`) are to be recognized while analyzing the source code.

**Traditional**

It determines if the pre-ANSI preprocessor is to be used. For the traditional preprocessor, a comment is equivalent to nothing, while for the ANSI preprocessor it is a white space. The traditional preprocessor does not delete comments within the `#define` directive, it does not recognize `"#"` and `"##"` operators, and requires an initial `"#"` in a directive at the very beginning of the line. It allows specification `foo()` if a macro `foo` takes a single argument, and permits a macro to be expanded recursively.

**Ignore #line**

It determines that the `#line` preprocessor directive be ignored.

## **2.5 Configuring Compiler Dialect and Header Directories**

An additional path configuration must be performed to specify the location of the HIWARE HC05 V5.x C compiler library header file (needed for DAC symbol analysis). On the **Options** menu, click **Compiler** to show the **Compiler Options** dialog box. This dialog box contains options which enable you to set C source code analysis parameters:

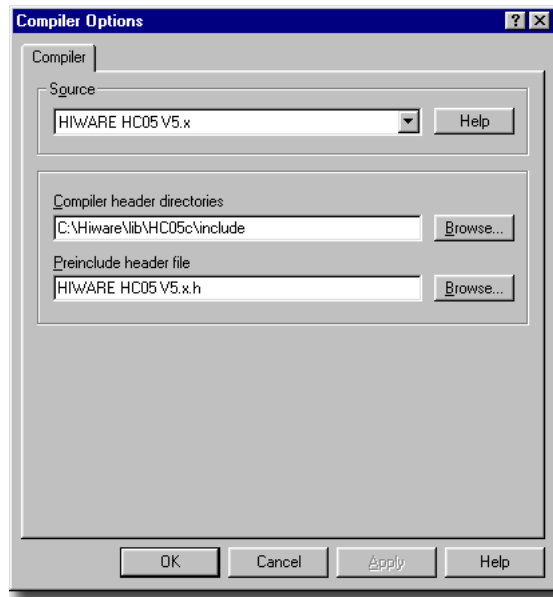


Figure 2.6 Compiler Options

### Source

The supported C dialects of the C language used in the active project can be selected in this box. In the "Demo" example project the HIWARE HC05 V5.x dialect is used.

### Compiler header directories

It enables you to set the list of directories which are to be searched for the files named using the "#include" directive. Different directories within a list are separated by a semicolon character. Only the listed directories are searched for files whose names are written enclosed in brackets ("<" and ">"). The search for files whose names are written enclosed in quotation marks ("") starts in the folder in which the source file containing the "#include" directive resides.

The list of header directories can be assigned to a file. In that case, the **Compiler header directories** box contains the file name (with absolute or relative path in relation to the Project Root) with the @ prefix. The directories listed in the file are separated by a semicolon or a new line (no semicolon necessary). The prefix @ is also used in the Project file ".dcp" to specify included files. Therefore, you should avoid using the character @ at the beginning of the names of files and folders that are to be used in the project.

The path of the Compiler libraries for the used CPU should be defined as the "*lib\HC05c\include*" subfolder of the HIWARE folder.

### Preinclude header file

It enables you to set the name of the file which will be included automatically at the beginning of every C source module during the analysis, as if the `#include` directive containing the name of the file in question enclosed in quotation marks (") was present in the first line of the analyzed source code. The preinclude file is used to specify predefined macros, variable and function declarations for a particular compiler which are not set by default in DAC analysis. The one corresponding to the "Demo" example project has been selected here: *"HIWARE HC05 V5.x.h"*. This file contains declarations of HIWARE HC05 V5.x C Compiler predefined macros. It is located in the *"Preinclude Files"* folder of the DAC folder. To adapt this file to your needs, on the **File** menu, point to **Configuration Files** and click **Compiler Preinclude File**.

## 2.6 Configuring Assembler Dialect and Header Directories

On the **Options** menu, click **Assembler** to open the **Assembler Options** dialog box. This dialog box contains options which enable you to set assembler source code analysis parameters:

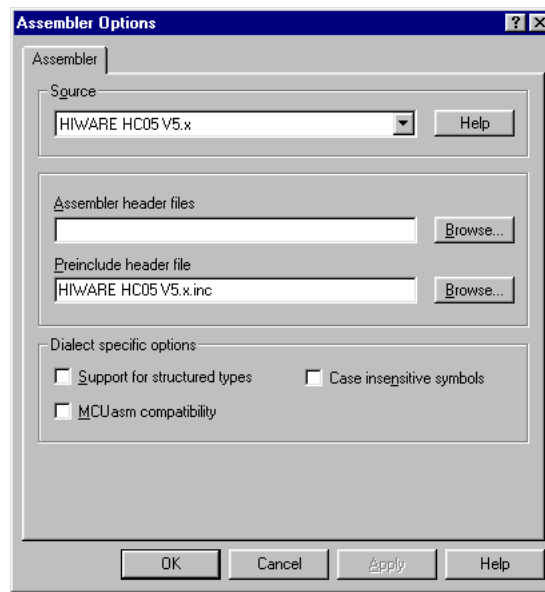


Figure 2.7 Assembler Options

### Source

It contains the list of supported assembler languages for the current project. It is possible to select a particular dialect, or the value **<none>** which means that parts of the C code containing inline assembler code will be ignored (information on symbols will not be retrieved, but no syn-

tax errors will occur either). In the "Demo" example project, the HIWARE HC05 V5.x assembler is used.

### **Assembler header files**

It determines a list of directories which are to be searched for the files named within the `INCLUDE` directive. Directories are separated by a semi-colon.

### **Preinclude header file**

If a file is specified in this edit box (typed in or selected using the **Browse** button), it will be included in each assembler file prior to commencing assembler analysis. This is identical to entering the `INCLUDE` directive in each assembler file.

### **Support for structured types**

It allows using the directives for dealing with structured types. Also, it allows using structured types in the assembly code.

### **MCUasm compatibility**

It includes MCUasm compatibility. In case this option is enabled, the label must end with a colon even if it begins in the first column.

### **Case insensitive symbols**

Enables case insensitive mode for symbol names. When enabled, all characters are converted to uppercase.

## **2.7 Adding Files to the Project**

In the **Project Window** the **Explorer View** replaces the Windows Explorer and supplies you with plenty of information on directories containing project files. It also gives you the possibility of adding files to the project. All files needed to run the "Demo" example will now be added to the project.

In the **Explorer View**, browse through the "*demo\HC05*" subfolder of the HIWARE folder, right-click on the "*Sources*" subfolder and from the shortcut menu choose **Add to project**. All file types configured in the previous section [Configuring File Types](#), are now added to the project:



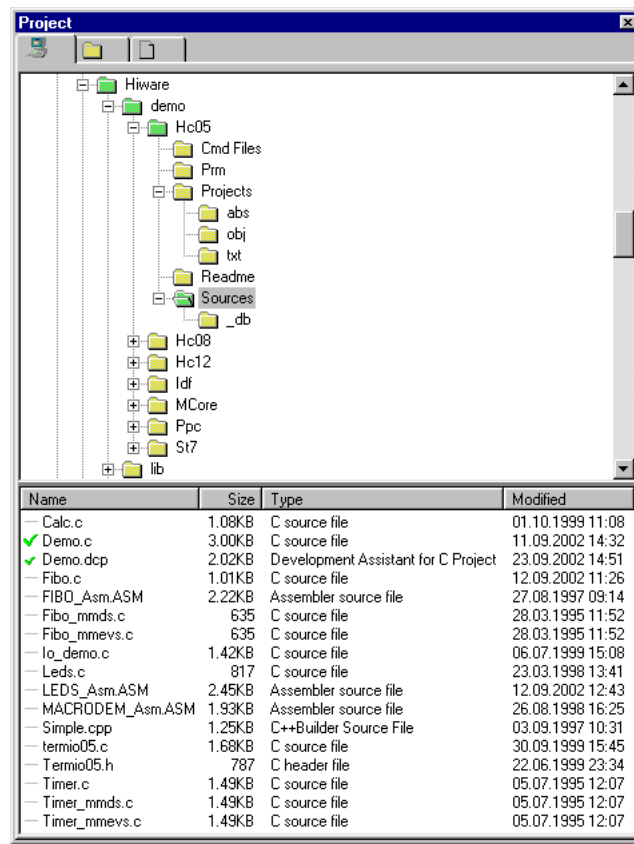


Figure 2.8 Project Window - Explorer View (Undocked mode)

Another alternative is to add each file separately by selecting each one and adding it to the project.

## 2.8 Building the Database

Development Assistant for C provides the static code analysis of C and ASM source files, and generates various data based on the results.

The analysis of the project source files and the generation of the database are divided into two phases: the analysis of individual program modules and the generation of data on global symbols usage. The results of the analysis are saved in database files on the disk, allowing them to be used in DAC later on. You can choose between the unconditional analysis of all project files and the analysis of changed source files only, using the **Build Database** and the **Update Database** commands from the **Start** menu, respectively. The **Update Database** command will optionally check if the include files used in program modules have been changed as well.

To build the database in the "Demo" example, on the **Start** menu click the **Build Database** command. This command performs the unconditional analysis of all project files and creates a database containing information on the analyzed source code. The errors and warnings detected during this

operation are displayed in the **Messages** window as illustrated below (for the "Demo.c" sample file):

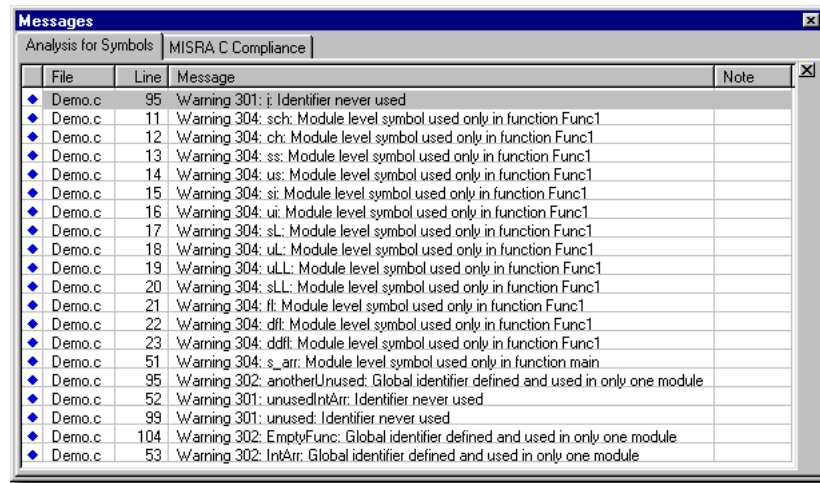


Figure 2.9 Messages Window - Analysis for Symbols tab (Undocked mode)

Once the analysis of all project files has been completed, the new database file containing the information on global symbols is created. Please refer to the DAC manual for further information on how symbols information can be used.

In the DAC **Project Window**, select the **Logical View** tab and unfold fields to get an overview of your project:

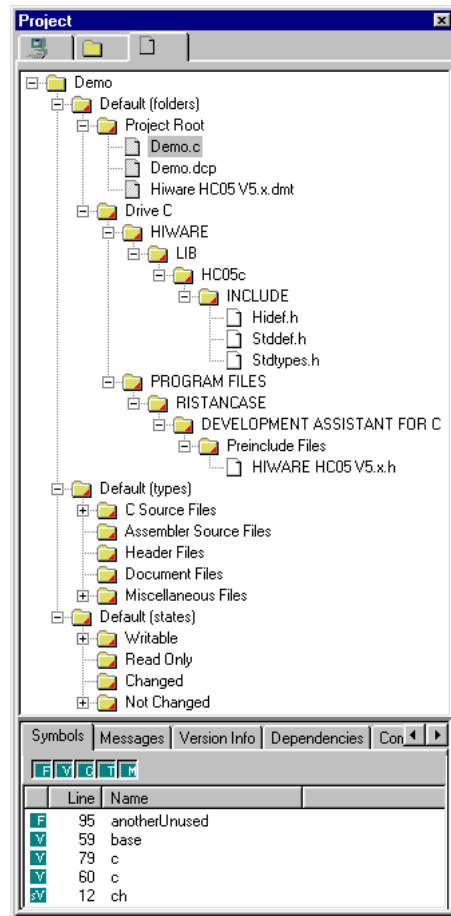


Figure 2.10 Project Window - Logical View (Undocked Mode)

---

## 3 Integrating the Tools

---

### 3.1 User-Defined Actions (UDA) Setup

To integrate the HIWARE HC05 V5.x tools and GNU make into DAC, on the **Option** menu, click **User-Defined Actions**, and then click the **Start Menu Actions** tab.

Now check the **Use compiler template** check box. By selecting this option, you have decided on what is probably the most common manner of using HIWARE HC05 V5.x tools. The action is contained in the *"HIWARE HC05 V5.x.tpl"* file located in the *"UDA Templates"* subfolder of the DAC folder. You can edit it and adjust it to your needs.

RistanCASE would appreciate your sharing additional features and ideas with us. Now click **OK**.

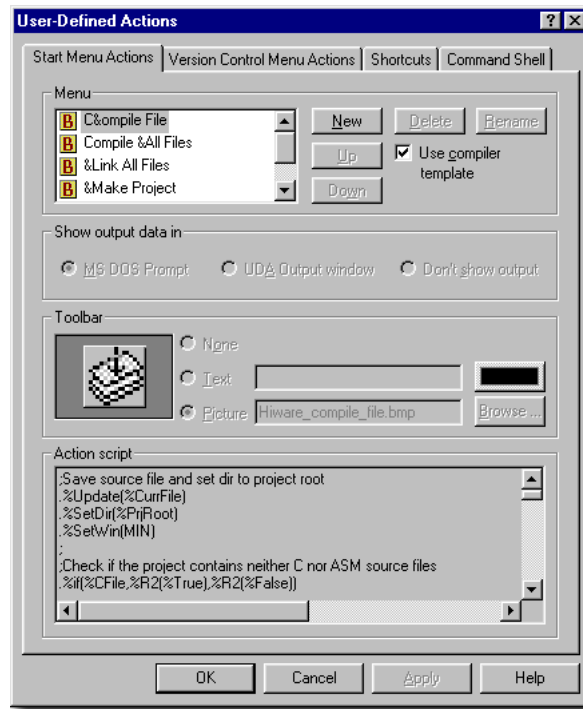







Figure 3.1 User-Defined Actions - Start Menu Actions

On the **View** menu, point to **Toolbars**, and then click **User-Defined Actions**. Five UDA buttons should now appear on the DAC toolbar, and the items **Compile File**, **Compile All Files**, **Link All Files**, **Make Project**, and **Build Project** on the **Start** menu. On the toolbar, appearing from left to right, the UDA buttons are:



Figure 3.2 UDA toolbar (undocked mode)

- Compile File 
- Compile All Files 
- Link All Files 
- Make Project 
- Build Project 

## 3.2 Makefile Template Setup

In order to completely define the **Compile File**, **Compile All Files**, **Link All Files**, **Make Project**, and **Build Project** UDA with the gmake utility, a project makefile must be provided. This file should have the same name

as the current project and the extension *".mak"*. It can be edited manually or generated automatically.

The template files are generally used to define all the necessary macros, commands, rules, and so on, which facilitate automatic project makefile generation by means of the Makefile Template Macro Language instructions expansion.

#### INFO:

You inevitably have to make the necessary customizations of the appropriate file for your compiler. Follow the directions given in the template file and consult your compiler tools literature if and when necessary.

Since manual editing is a very tedious and error-prone process, an example makefile template *"HIWARE HC05 V5.x.dmt"* has been provided in the *"Makefile Generator Templates"* subfolder of the DAC folder.

Another file that should be customized to reflect the properties of the target system on which a program is to be executed is the linker command file. If the *"HIWARE HC05 V5.x.dmt"* makefile template is expanded, the linker command file will automatically be generated along with the makefile. The template file section responsible for the linker command line file generation is at the end of the template and is commented.

To enable automatic makefile generation, do the following:

- Copy the file *"HIWARE HC05 V5.x.dmt"* from the *"Makefile Generator Templates"* subfolder of the DAC folder to the *"demo\HC05\Sources"* subfolder of the the HIWARE folder.
- Customize *"HIWARE HC05 V5.x.dmt"* to your needs.
- On the **Options** menu, click **Makefile Generator**, and then select **Generate makefile or its components**.
- In the **Makefile template** box enter *"HIWARE HC05 V5.x.dmt"* or use the **Browse** button to find it.
- Click **OK**.
- On the **Start** menu, click **Generate Makefile**. The makefile for your project will be generated automatically.

#### NOTE:

On every start of the Build or Update the Database command from the Start menu, DAC will automatically check if the makefile needs to be regenerated.

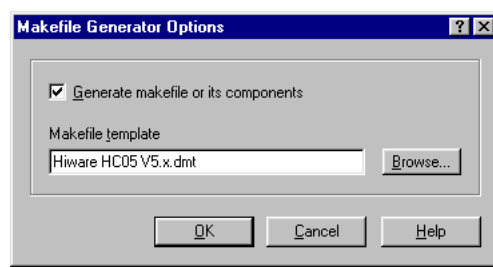


Figure 3.3 Makefile Generator Options

The report of the automatic project makefile generation process is displayed in the **Messages** window under the **Makefile Generator** tab. After

the project makefile has successfully been created, you can use the **Link All Files**, **Make Project**, or **Build Project** UDA to generate the *"Demo.abs"* file.

The generated *"Demo.mak"* file can be opened for inspection. Please, bear in mind that any changes made in it would be overwritten as soon as any of the following actions from the **Start** menu is performed: **Build Database**, **Update Database** or **Generate Makefile**. This means that if any changes to makefiles are to be made, the make template file should be edited instead.

DAC UDA handles compiler error reports and displays them in the **Messages** window under the **Compiler Messages** or **Linker Messages** tab, depending on the UDA performed.

### **3.3 Using HIWARE Tools by Default**

If you want your newly created project files to automatically inherit the proper setup for HIWARE HC05 V5.x tools, you should also customize the default DAC project file *"dac.dcp"*, which is located in the *"Program"* subfolder of the DAC folder.

---

## 4 Limitations of Static Code Analysis

---

### 4.1 Analysis for Symbols, C Source

Since DAC Static Code Analysis supports a great number of dialects, it stands to reason that each and every one of these dialects cannot be supported in minute detail.

However, if you find that any of these limitations prevents you from using DAC in the optimal way, please, send us the description of that limitation to: [support@RistanCASE.com](mailto:support@RistanCASE.com).

You can find more about these limitations in User Manual, chapters: Pre-processor, and Dealing with Syntax Extensions.

### 4.2 Analysis for Symbols, ASM Source

HIWARE assemblers have support for the structured types.

If the [Support for structured types](#) option is selected, the following directives will be analyzed: **struct**, **union**, **endstruct**, **endunion**, **type**. If this option is not selected, an error will be reported in case any of the mentioned directives is used.

All HIWARE assemblers also have the support for embedding inline assembler blocks in C code. There are two syntaxes for inline assembler blocks:

```
asm assembler_instruction
```

and



```
asm
{
    /* assembler block */
}
```

Inline assembler blocks analysis is supported for both the syntaxes. Also the specific details in which the code within inline assembler blocks differs from the assembler code in separate files are supported. Since these specifics differ depending on particular assemblers, they will be listed within their specific groups.

The support for using the pointer in the assembler is provided on the level where the following is realized:

in case of correct usage	no error is reported
in case of incorrect syntax	an error is reported, but no further check is carried out. For example in case of " a -> b " expression, there is no check whether " a " is a structural type, and whether it has the field " b ". If the register is used as a pointer, it is also not checked whether it points to the variable of the appropriate type.
pointer definitions	pointer definitions are supported through the "pseudo opcode" <b>pointer</b> .

All HIWARE assemblers allow using operation extensions ( **.b**, **.w** and **.l**, for example, **mov.b**) in inline assembler blocks.

#### NOTE:

HIWARE inline assembly syntax does not allow that a period character is part of an identifier name.

HIWARE assembler syntax (which is applied to separate assembler files) allows the " ." character to be part of an identifier name. For this reason, it is not advisable to give an identifier a name ending with **.b**, **.l** or **.w**. A name ending such as this one can, depending on the context, be understood as a mark of operation length, and be discarded in the process of analysis.

### 4.2.1 Notes about assembler support for the specific HIWARE assemblers

*HIWARE HC05 V5.x, HC11 V5.x, HC16 V5.x, M68xxx/CPU32 V5.x, H8/500 V5.x, XA V5.x*

**In the inline mode:**

Label	In the inline mode the label beginning in the first line does not have to end with a colon, unless the option <a href="#">MCUasm compatibility</a> is selected. Within inline assembler blocks labels must end with a colon.
Allowed directives	<b>dc, dcb, dcw and dcl.</b> If you try to use other directives, an error will be reported.

*HIWARE HC08 V5.x and M-CORE V5.x*

**In the inline mode:**

Label	In the inline mode the label beginning in the first line does not have to end with a colon, unless the option <a href="#">MCUasm compatibility</a> is selected.
Allowed directives	<b>dc, _assert, _stack, _adj, _ingen, dcb, dcw, dcl, dcf, dcd.</b>

*HIWARE ST7 V5.x*

**In the inline mode:**

Label	In the inline mode the label beginning in the first line does not have to end with a colon, unless the option <a href="#">MCUasm compatibility</a> is selected.
Allowed directives	If the option <b>ST7 compatibility</b> is selected, expression syntax differs from standard syntax. See <i>HIWARE ST7 Reference Manual</i> . ST7 compatibility is valid only for separate files, that is, not for inline assembler blocks.

**HIWARE HC12 V5.x****In the inline mode:**

---

Label	In the inline mode the label beginning in the first line does not have to end with a colon, unless the option <a href="#">MCUasm compatibility</a> is selected.
Allowed directives	If the <b>Semi-Avocet compatibility</b> option is selected, the following directives can be used: <b>switch</b> , <b>case</b> , <b>endsw</b> , <b>default</b> . Otherwise, the <b>switch</b> directive is used for choosing some of the previously defined assembler sections, and if you try to use some of the other directives, an error is reported.

---

---

## 5 Debugger

---

**NOTE:**

This section of the technical note only provides information on how to configure DAC to facilitate your work with the HI-WAVE debugger V6.0, for which the document has primarily been written. Using it as a basis, DAC can easily be configured to work with other HIWARE tools. For further information on DAC, please refer to "Development Assistant for C" documentation V4.0.

DAC V4.0 is currently integrating a **DAPI** (Debugging support Application Programming Interface). This interface will enable DAC to exchange messages with HI-WAVE. The advantages of such a connection are that it allows you to set or delete break points from within DAC and to execute other debugger operations. Furthermore, DAC is not only synchronized with HI-WAVE, but runs with it as it operates - it is always in the same file and on the same line as HI-WAVE. Thus, the usability of both DAC and HI-WAVE is increased.

To make the most of your HI-WAVE debugger V6.0 with DAC, these are the steps to follow:

- [Installation of communication DLL's](#)
- [Configuring DAC Debugger Options](#)
- [Configuring HI-WAVE Debugger Options](#)
- [Synchronized Debugging](#)
- [Troubleshooting](#)

An example configuration for the M68k CPU, which can easily be adapted to each CPU supported by HIWARE, can be found on RistanCASE's CD-ROM in the "DAC\Technical Notes\HIWARE" folder, or downloaded from RistanCASE's web site:

<http://www.RistanCASE.com/dac/v40/specificsupport/metrowerks.php>

### 5.1 Installation of communication DLL's

HI-WAVE and DAC communication is based on DDE protocol. The whole system contains:

- HI-WAVE
- DAC

- Generic DAPI interface implementation DLLs - used by DAC for DDE communication ("*CDHIWAVE6x.dll*" and "*DnDAPI32.dll*")
- A DAPI communication DLL - used by HI-WAVE DLL ("*Ndapi.dll*")
- A HI-WAVE specific DLL - for bringing its interface to the debugging environment and DAC's DAPI ("*Dac.wnd*", provided by HIWARE)

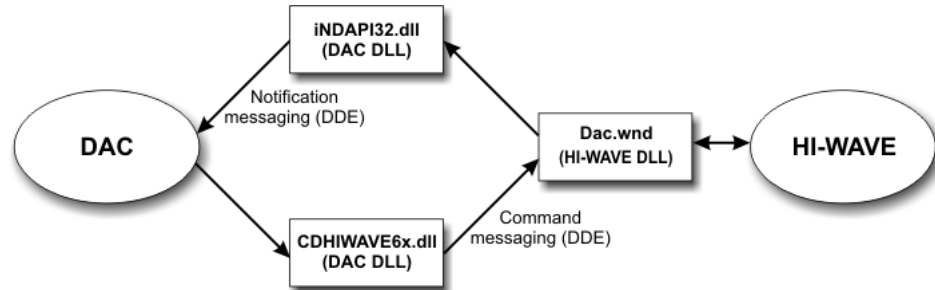


Figure 5.1 Communication between DAC and HI-WAVE

The files "*CDHIWAVE6x.dll*" and "*DnDAPI32.dll*" will be installed automatically during the DAC installation process.

As previously described, HI-WAVE needs the DAPI communication DLL (provided by DAC). This DLL ("*Ndapi.dll*") is automatically installed during the HIWARE toolkit installation process. However, if you are installing a new DAC release (later than V3.5, build 555), you will have to follow this procedure:

- From the "*Program*" subfolder of the DAC folder, copy "*iNdapi32.dll*" (version 3.5.555 or later) into the "*PROG*" subfolder of the HIWARE folder (in which HI-WAVE is located). Then rename "*iNdapi32.dll*" to "*Ndapi.dll*".

## 5.2 Configuring DAC Debugger Options

On the **Options** menu, click **Debugger** to open the **Debugger Options** dialog box, which has four tabs [Debugger](#), [Connection](#), [Project Dependent](#), and **About Plug-In**.

### 5.2.1 Debugger

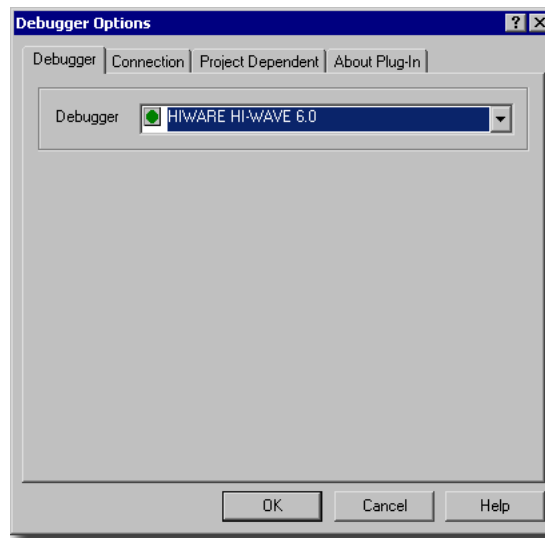


Figure 5.2 Selecting the Debugger

In the Debugger field, select the corresponding debugger: "HIWARE HI-WAVE 6.0".

### 5.2.2 Connection

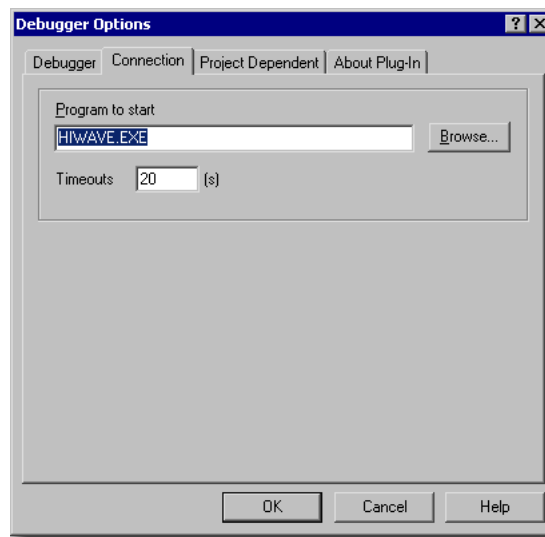


Figure 5.3 Debugger Options - the Connection tab

#### Program to start

In the **Program to start** box, specify the path and name of the HI-WAVE executable. (Default: HIWAVE.EXE)

## Timeouts

During establishing the connection, DAC extensively communicates with the external debugger, sending and receiving data. Depending on different technologies, data transfer can be faster or slower, and sometimes DAC lets the debugger have the control, waiting for an answer. In the whole communication, timeouts are often used to insure that establishing and maintaining the connection will be carried out in the foreseeable time. The length of timeouts depends on the selected debugger.

### 5.2.3 Project Dependent

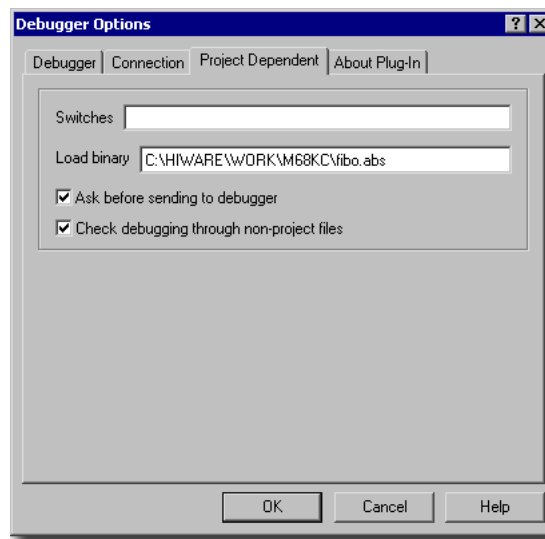


Figure 5.4 Debugger Options - the Project Dependent tab

## Switches

Leave it empty.

## Load binary

Specify the binary file to be opened (debugged). In the example provided for your use, the "*fibo.abs*" file is to be debugged.

## Ask before sending to debugger

If this option is selected, before establishing the connection between DAC and the debugger, you will be asked to confirm initialization of the debugger after the connection is established.

The debugger initialization is a debugger specific process, which can discontinue previous debugging. Therefore, you will probably turn on **Ask before initializing debugger** option if you have already started debugging process and want to continue that process with DAC where you left in debugger.

### Check debugging through non-project files

During the debug session in the external debugger, simulation can stop somewhere in the code. DAC, naturally, follows the point where the simulation has stopped, and represents it in the appropriate way in its environment. If the simulation has stopped in the code that is not in the DAC project, DAC will not be able to offer complete functionality. If you select this option, DAC will follow debugging and notify you in case it leaves the DAC project, so that you will be able to update the project and make the most of DAC features.

## 5.3 Configuring HI-WAVE Debugger Options

Before configuring the project file, please, quit DAC.

- Open HI-WAVE and on the **File** menu click **Open Project**. Select the "Project.ini" file from the currently defined working directory (in this example it is "*WORK\<processor>c\*" subfolder of the HIWARE folder).
- In the project layout, the HI-WAVE DAC component "*dac.wnd*" will now be added. In HI-WAVE, on the **Component** menu, click **Open**, and then, choose **Dac**:

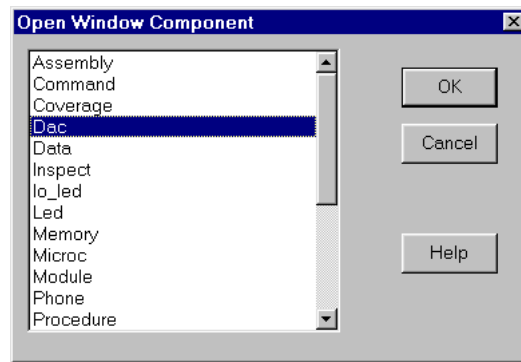


Figure 5.5 Open Window Component

The HI-WAVE DAC window, needed for communication with DAC, is now open:



Figure 5.6 DAC Link

- On the **File** menu, in HI-WAVE, click **Save Project**, to save this configuration. Thus this component will be loaded automatically next time this project is called.



- Quit HI-WAVE.

## 5.4 Synchronized Debugging

It is now possible to test the synchronization between DAC and HI-WAVE.

- Open the project in DAC.
- Open "*Fibo.c*" if it's not already open.
- Right-click the "*Fibo.c*" source window and select **main** from the shortcut menu. The pointer now points to the "`void main(void) {`" statement.
- In DAC, on the **Debug** menu, click **Set Breakpoint**, or click the corresponding button on the **Debug** toolbar ([Figure 5.7](#).)

The selected line is marked in configurable color, indicating that a breakpoint has been set.

- On the **Debug** menu, click **Run**.

HI-WAVE has now been started and after a certain period of time, it will stop at the specified breakpoint.

### NOTE:

In case of changes in your source code, don't forget to rebuild the Database when generating new binary files to avoid misalignment between HI-WAVE and DAC source positions.



Figure 5.7 Toolbar with the debugger commands (from left): Synchronize, Run, Stop, Step Into, Step Out, Step Over, Go Until, Set Breakpoint, Clear Breakpoint.

Starting now, you can debug directly from DAC using the toolbar, or even from HI-WAVE. Synchronization between the two programs has been ensured.

## 5.5 Troubleshooting

Possible problems and ways to overcome them:

<i><b>Problem</b></i>	<i><b>Answer</b></i>
<i>There is no "HIWARE HI-WAVE 6.0" in the <b>Debugger</b> box on the <b>Debugger Options</b> dialog box (<a href="#">5.2.1 "Debugger"</a>).</i>	<p>Check DAC Version and release number (On the <b>Help</b> menu, click <b>About</b>). It should be V4.0, build 055 or later. Check the "Program" subfolder of the DAC folder for necessary DLL. The required DLL is: "CDHIWAVE6x.dll".</p> <p>Probably, HI-WAVE support in DAC has not been licensed as well. If this is the case, please, contact your supplier.</p>
<i>When loading the DAC component into HI-WAVE, the message box entitled <b>Component Loading Error</b> is displayed with the following message: "Could not load component "C:\HIWARE\PROG\dac.wnd" or with one of its libraries. (Error code 1157)"</i>	<p>Check if the "Ndapi.dll" is located in the "prog" subfolder of the HIWARE folder. If not, copy the specified DLL into it, as described in section <a href="#">5.1 "Installation of communication DLL's"</a>.</p>

**Problem**

*DAC displays the alert message box with the message "External debugger tried to connect to a DAC project dissimilar to the one configured!".*

**Answer**

This means that the name currently specified in the **Debugger Options** dialog box does not match the debugger name specified in HI-WAVE.

Open the **Setup** dialog box in HI-WAVE by clicking the **DAC Link** component and choosing the **Setup** command from the **DAC Link** menu.

The [Connection Specification](#) dialog box is opened ([Figure 5.8](#)).

Compare the **Debugger Name** in this dialog box with the selected [Debugger](#) in DAC ([Figure 5.2](#)).

They must be identical.

If this is not the case, change the debugger name in the HI-WAVE **Connection Specification** dialog box and click **OK**.

This will cause a new connection to be established and the connection specification to be saved in the current "*Project.ini*" file in the following section:

```
[DAC]
DEBUGGER_NAME=HIWARE HI-WAVE 6.0
SHOWPROT=1
```

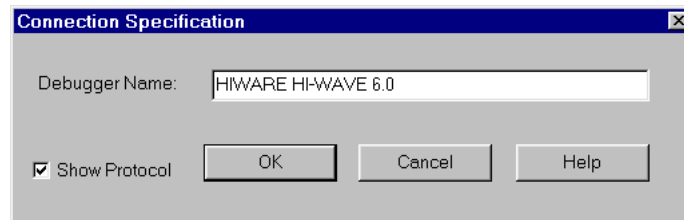


Figure 5.8 Connection Specification dialog box



## 6 Index

### Symbols

#include 8, 10  
 .dcp 3  
 @ 8  
 @ prefix 10  
 \_db 4

### A

Analysis for Symbols, ASM Source 20  
 Analysis for Symbols, C Source 20

### B

Build Project 17, 19  
 Building the Database 13

### C

C Source 8  
 C:\Hiware\demo\HC05 1  
 CDHIWAVE6x.dll 25, 30  
 Compile All Files 17  
 Compile File 17  
 Compiler  
   *header directories* 10  
   *Help* 5  
   *libraries* 10  
 Configuring  
   *Analysis for Symbols* 6  
   *Assembler Dialect and Header Directories* 11  
   *Compiler Dialect and Header Directories* 9  
   *DAC* 2  
   *File Types* 5  
   *Working Directories* 3

### D

-D 8  
 -d 8  
 DAC  
   - *V4.0.055* 1  
   *Demo Mode* 1  
   *toolbar* 17  
 dac.dcp 19  
 Dac.wnd 25  
 dac.wnd 30  
 DAPI 25  
 DAPI (Debugging support Application Programming Interface) 24  
 Database 4  
   *Build* 13, 19  
   *directory* 4  
   *Update* 13, 19  
 Debugger 24  
 Defines / Control file 8  
 Demo 2-3, 10-13  
 Demo.abs 19  
 Demo.c 14  
 Demo.dcp 3  
 Demo.mak 19  
 DnDAPI32.dll 25

### E

embedding inline assembler blocks 20  
 Explorer View 12

### F

File Types tab 5  
 Free Software Foundation 1

---

**G**

General 7  
Generate Makefile 19  
Generate makefile or its components 18  
gmake 1  
GNU 1

---

**H**

HIWARE  
    *HC05 V5.x* 1, 5, 10–11, 16, 19  
    *Windows Workbench Help* 5  
HIWARE HC05 V5.x.dmt 18  
HIWARE HC05 V5.x.tpl 16  
HIWARE HI-WAVE 6.0 26, 30–31  
HIWARE inline assembly syntax 21  
HI-WAVE debugger V6.0 24  
HIWAVE.EXE 26  
hiwave.hlp 5

---

**I**

-I 8  
-i 8  
iNdapi32.dll 25  
Integrating the Tools 16

---

**L**

license  
    *Demo Mode* 1  
    *trial* 1  
Limitations of Static Code Analysis 20  
Link All Files 17, 19  
Logical View 15

---

**M**

M68k CPU 24  
Make Project 17, 19  
Makefile  
    *Generator* 18  
    *Generator Templates* 18  
    *Template Setup* 17  
Maximum identifier length 8  
Messages window 14

---

**N**

Ndapi.dll 25, 30  
New Project 3

---

**O**

Options 18

---

**P**

prefix @ 8  
Preinclude file 11  
Project  
    *Adding Files to the Project* 12  
    *Creating a New Project* 3  
    *Root Directory* 4  
    *Window* 12, 15

---

**R**

Referential 4  
Requirements 1  
RistanCASE's CD-ROM 2, 24

---

**S**

Semi-Avocet compatibility 23  
Software Metrics 4  
Source 10  
Special table processing 8  
ST7 compatibility 22  
Start 13, 17  
Start Menu Actions 16  
Symbols 4

---

**T**

TEXTPATH environment variable 5

---

**U**

UDA  
    *buttons* 17  
    *Toolbar* 17  
Use compiler template 16  
User  
    *Defined Actions (UDA) Setup* 16  
    *Help File* 5  
Using HIWARE Tools by Default 19  
using the pointer in the assembler 21